Attorney Docket No. 225676
MS # 306066.01

PATENT APPLICATION

Invention Title:

METHOD FOR EFFICIENT CONTENT DISTRIBUTION USING A PEER-TO-PEER NETWORKING INFRASTRUCTURE

Inventors:

Todd R. Manion	US	Redmond	Washington
INVENTOR'S NAME	CITIZENSHIP	CITY OF RESIDENCE	STATE or FOREIGN COUNTRY
Ravi T. Rao	Canada	Redmond	Washington
INVENTOR'S NAME	CITIZENSHIP	CITY OF RESIDENCE	STATE or FOREIGN COUNTRY
Michael Shappell	US	Issaquah	Washington
INVENTOR'S NAME	CITIZENSHIP	CITY OF RESIDENCE	STATE or FOREIGN COUNTRY
INVENTOR'S NAME	CITIZENSHIP	CITY OF RESIDENCE	STATE or FOREIGN COUNTRY
INVENTOR'S NAME	CITIZENSHIP	CITY OF RESIDENCE	STATE or FOREIGN COUNTRY
INVENTOR'S NAME	CITIZENSHIP	CITY OF RESIDENCE	STATE or FOREIGN COUNTRY

Be it known that the inventors listed above have invented a certain new and useful invention with the title shown above of which the following is a specification.

METHOD FOR EFFICIENT CONTENT DISTRIBUTION USING A PEER-TO-PEER NETWORKING INFRASTRUCTURE

TECHNICAL FIELD

[0001] The present invention relates generally to the distribution of content in a computer network and, more particularly, to methods for efficiently distributing content using a peer-to-peer networking infrastructure.

BACKGROUND OF THE INVENTION

[0002] Traditionally, the task of distributing content over a network to a client computing device has been one that is highly server-intensive. In the typical model, the clients will connect to a server to receive content directly from the server on an individual basis. In the case of a large enterprise installation, for instance, this model implies that every client is required to connect to the server to receive data from it making it very difficult for the server to handle such large amounts of simultaneous requests for data.

[0003] A server-only content distribution model can also present problems with respect to bandwidth availability. With the advent of affordable and easily maintainable home networking equipment it is becoming more common for several computers in a household, or residential area, to share a single broadband or dial-up connection. A household having five computers, for example, may need to download software (e.g., patches, product upgrades, etc.) from a central server (e.g., windowsupdate.com) to each computer. Since there is a single point of receipt from outside the home network, if all five computers, or even less than all five, are requesting the same content at overlapping

times the computers will be splitting the available bandwidth of the connection while receiving the same content.

[0004] Not only does a server-based content distribution solution have drawbacks in the enterprise and home settings but also in the internet setting as well. For example, the interconnected nature of the Internet has, among other things, accelerated the spread of computer viruses and worms. Unfortunately, virus cleansing and repair remains a reactionary process whereby the necessary virus definition files are distributed upon identification of the virus "in the wild." Time is therefore of the essence in distributing the virus definition files to stanch the spread of the virus. When a new virus is first identified, the virus definition distribution servers can become overloaded with requests or could even be made unavailable (e.g., through a denial of service attack or some similar nefarious method) as part of the scheme to propagate the spread of the virus. A solution where the virus definitions are obtained only from a centralized server on the internet fails to safeguard against this eventuality and additionally fails to provide a method whereby the virus definitions can be distributed to the maximum number of computers in the most efficient fashion.

[0005] One potential solution to the problems described above is to use a series of redundant content distribution servers that may serve to distribute the load of demand over a number of servers. Such a solution however has several drawbacks. First, server hardware and software, and in particular the type of server hardware and software needed for intensive data delivery tasks, is typically expensive and requires experienced administration resources. Additionally, such a solution only scales in a linear fashion. For example, suppose 1,000 clients are currently receiving their content from a single

server. Adding one more content distributing server reduces the average number of clients to a server to 500. Adding a third reduces this number to approximately 333, and so on. Thus a significant number of servers must be added to reduce the number of clients receiving content from a particular server to desired or manageable levels. Finally, a further drawback to this solution is that the content to be distributed and the distribution ability will always remain solely on the servers and hence only available in a limited fashion.

SUMMARY OF THE INVENTION

[0006] In view of the foregoing, the present invention provides a method for efficiently distributing content by leveraging the use of a peer-to-peer network infrastructure. Peer-to-peer networking provides an infrastructure that enables computing devices to communicate and share information securely with one another without the need of a server or other third party to facilitate the communication. A peer-to-peer networking infrastructure can be effectively employed to improve the efficiency of content distribution and the corresponding scalability. In a network of peers, a handful of the peers can receive content from centralized servers. These peers can then flood this content out to a few more peers who in turn can send the content along to others. Ultimately, this method produces a result whereby a request for content can be fulfilled by locating the closest peer and obtaining the content from that peer.

[0007] In one embodiment the above method can be used to distribute content over the Internet or within an enterprise installation by creating content distribution groups of one or more computing devices that are also peers in one or more peer-to-peer networks.

Content requests on the server can then be redirected to a content distribution group for distribution to peers to reduce load on the centralized content distribution server.

[0008] A further contemplated embodiment efficiently streams time sensitive content through the use of a spanning tree architecture of peer-to-peer clients. On-line meeting materials or webcast concerts or similar time sensitive content can be distributed in a highly efficient manner by geometrically increasing the amount of content distributing computing devices over time.

[0009] In yet another embodiment the present invention provides for more efficient use of bandwidth in home networks or shared residential broadband or dial-up connections. In a peer-to-peer content distribution scenario one client computing device can download content over the connection while the other computing devices can simply obtain the content from the peer that downloaded it, thereby eliminating the need for bandwidth to be used by multiple client computing devices in downloading the same content.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] While the appended claims set forth the features of the present invention with particularity, the invention, together with its objects and advantages, may be best understood from the following detailed description taken in conjunction with the accompanying drawings of which:

[0011] Figure 1 is a schematic diagram of an exemplary computer architecture on which the method of the invention can be implemented;

[0012] Figure 2 is a schematic diagram showing an exemplary communications network in which the method of the invention can be practiced;

[0013] Figure 3 is a schematic diagram showing an exemplary peer-to-peer networking infrastructure architecture;

[0014] Figure 4 is a schematic diagram illustrating the process of a peer node joining a peer-to-peer group;

[0015] Figure 5 is a flowchart illustrating the process of a peer node joining a peer-to-peer group;

[0016] Figure 6 is a schematic diagram showing an exemplary peer-to-peer group;

[0017] Figure 7 is a flowchart illustrating the process of content distribution leveraging a peer-to-peer networking infrastructure;

[0018] Figure 8 is a schematic diagram showing an implementation of content distribution leveraging a peer-to-peer networking infrastructure for a residential network;

[0019] Figure 9 is a schematic diagram showing an implementation of content distribution leveraging a peer-to-peer networking infrastructure for the Internet; and

[0020] Figure 10 is a schematic diagram showing an implementation of content distribution leveraging a peer-to-peer networking infrastructure for an enterprise installation.

DETAILED DESCRIPTION OF THE INVENTION

[0021] In the description that follows, the invention is described with reference to acts and symbolic representations of operations that are performed by one or more computing devices, unless indicated otherwise. As such, it will be understood that such

acts and operations, which are at times referred to as being computer-executed, include the manipulation by the processing unit of the computing device of electrical signals representing data in a structured form. This manipulation transforms the data or maintains them at locations in the memory system of the computing device, which reconfigures or otherwise alters the operation of the computing device in a manner well understood by those skilled in the art. The data structures where data are maintained are physical locations of the memory that have particular properties defined by the format of the data. However, while the invention is being described in the foregoing context, it is not meant to be limiting as those of skill in the art will appreciate that several of the acts and operations described hereinafter may also be implemented in hardware.

[0022] Turning to the drawings, wherein like reference numerals refer to like elements, the invention is illustrated as being implemented in a suitable networking environment. The following description is based on illustrated embodiments of the invention and should not be taken as limiting the invention with regard to alternative embodiments that are not explicitly described herein.

I. Exemplary Environment

[0023] Referring to Figure 1, the present invention relates to communications between network nodes on connected networks. Each of the network nodes resides in a device that may have one of many different computer architectures. For descriptive purposes, Figure 1 shows a schematic diagram of an exemplary architecture usable for these devices. The architecture portrayed is only one example of a suitable environment and is not intended to suggest any limitation as to the scope of use or functionality of the

invention. Neither should the computing devices be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in Figure 1. The invention is operational with numerous other general-purpose or special-purpose computing or communications environments or configurations. Examples of well known computing systems, environments, and configurations suitable for use with the invention include, but are not limited to, mobile telephones, pocket computers, personal computers, servers, multiprocessor systems, microprocessor-based systems, minicomputers, mainframe computers, and distributed computing environments that include any of the above systems or devices.

[0024] In its most basic configuration, a computing device 100 typically includes at least one processing unit 102 and memory 104. The memory 104 may be volatile (such as RAM), non-volatile (such as ROM and flash memory), or some combination of the two. This most basic configuration is illustrated in Figure 1 by the dashed line 106.

[0025] Computing device 100 can also contain storage media devices 108 and 110 that may have additional features and functionality. For example, they may include additional storage (removable and non-removable) including, but not limited to, PCMCIA cards, magnetic and optical disks, and magnetic tape. Such additional storage is illustrated in Figure 1 by removable storage 108 and non-removable storage 110. Computer-storage media include volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage of information such as computer-readable instructions, data structures, program modules, or other data. Memory 104, removable storage 108, and non-removable storage 110 are all examples of computer-storage media. Computer-storage media include, but are not limited to, RAM,

ROM, EEPROM, flash memory, other memory technology, CD-ROM, digital versatile disks, other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage, other magnetic storage devices, and any other media that can be used to store the desired information and that can be accessed by the computing device.

[0026] Computing device 100 can also contain communication channels 112 that allow it to communicate with other devices. Communication channels 112 are examples of communications media. Communications media typically embody computer-readable instructions, data structures, program modules, or other data in a modulated data signal such as a carrier wave or other transport mechanism and include any information-delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communications media include wired media, such as wired networks and direct-wired connections, and wireless media such as acoustic, radio, infrared, and other wireless media. The term computer-readable media as used herein includes both storage media and communications media. The computing device 100 may also have input components 114 such as a keyboard, mouse, pen, a voice-input component, and a touch-input device. Output components 116 include screen displays, speakers, printers, and rendering modules (often called "adapters") for driving them. The computing device 100 has a power supply 118. All these components are well known in the art and need not be discussed at length here.

II. Server-based Content Distribution

[0027] The present invention is directed to methods for efficiently distributing content over computer networks. Referring to Figure 2, an exemplary communications network architecture is illustrated. Accompanying a computing device 100 on a local area network (LAN) 120 is a server 200 and a router 202. The router 202 allows the devices on the LAN 120 to communicate over an internetwork 204 to remote computing devices 206. The Internet is one example of an internetwork. In the present example, the server 200 can be a network server which the computing device 100 may access for content distribution over the LAN 120 and the remote computing device can be a remote content distribution server which the computing device may access for content distribution over the Internet 204 or similar wide area network (WAN).

[0028] Traditionally, the task of distributing content over a network 120 to a client computing device 100 has been one that is highly server-intensive. In the typical model, the clients 100 will connect to a server 200, 206 to receive content directly from the server 200, 206 on an individual basis. In the case of a large enterprise installation, for instance, this model implies that every client 100 is required to connect to the internal network server 200 to receive content (e.g., data) from it making it very difficult for the server to handle large amounts of simultaneous requests for content. In the case of a home network or similar shared residential broadband or dial-up connectivity scenario, this model implies that every computer 100 sharing the connection 208 is required to connect to the external content server 206 to receive data from it, thereby splitting the available bandwidth of the connection 208 when requests are made by the computing devices 100 at overlapping times. External content servers 206 exposed on the Internet

can become overloaded when there is a spike in requests for time sensitive content regardless of whether the requests are originating from inside an enterprise installation or residential gateway. In addition, the server based content distribution model may fail to provide content availability contingency scenarios for periods of server unavailability. Attempts to address the limitations of server-based content distribution have focused on additional server redundancy, however such solutions do not scale well under severe demand and are generally not cost effective.

III. Peer-to-peer Networking Infrastructure

[0029] The present invention leverages the use of a peer-to-peer network infrastructure for efficient content distribution. In the description that follows the invention is described as being implemented over a peer-to-peer network infrastructure such as the Windows® Peer-to-Peer Networking Infrastructure by Microsoft of Redmond, Washington. As will be appreciated by one of ordinary skill in the art, the network infrastructure should be understood to include any network infrastructure possessing the necessary communications and security protocols.

[0030] Turning to Figure 3, an exemplary peer-to-peer architecture is illustrated. The peer-to-peer infrastructure resides primarily between the operating system (OS) 300 and socket 302 layers and the application layer 318. Of particular importance are the discovery 308, graphing 306, and grouping 312 components. At the bottom most layer of the architecture is the core OS 300. The OS is, in one embodiment, the Windows operating system by Microsoft of Redmond, Washington. As will be appreciated by one of ordinary skill in the art, the OS should be understood to also include similar operating

systems. Residing between the peer-to-peer architecture and the OS is the network communications socket layer 302. The socket layer 302 allows the peer-to-peer architecture to leverage the operating system's interface with lower level network communication protocols. The discovery protocol 308 is a peer name resolution protocol (PNRP) that enables the resolution of a peer name to an IP address in a completely serverless fashion. Thus, with the discovery protocol 308, it is possible for a computing device to locate a peer on a network without the need for a separate server to maintain peer network location information. The graphing component 306 allows for the organization of a given set of network nodes so that data can be efficiently disseminated amongst them. Specifically, graphing 306 addresses data dissemination in a network that is not fully connected and provides for one-to-many dissemination of the data. The grouping component 312 associates a security model with one or more graphs created by the graphing component 306. With the grouping 312 module, a peer is able to create a group, own a group, and define members (users or machines) that are allowed to join the group.

[0031] In one embodiment of the graphing component 306 employed by the present invention, interfaces and methods can be implemented through an application programming interface (API). Such an embodiment is particularly well suited for the Microsoft Windows XP operating system in which the APIs may be as follows:

```
// Graph interfaces

HRESULT WINAPI PeerGraphStartup(
                IN  WORD  wVersionRequested,
                OUT PPEER_VERSION_DATA pVersionData);

HRESULT WINAPI PeerGraphShutdown();

VOID WINAPI PeerGraphFreeData(
                IN  PVOID       pvData);
```

```
HRESULT WINAPI PeerGraphGetItemCount(
                IN  HPEERENUM   hPeerEnum,
                OUT PULONG      pCount);

HRESULT WINAPI PeerGraphGetNextItem(
                IN  HPEERENUM   hPeerEnum,
                IN OUT PULONG   pCount,
                OUT PVOID     * ppvItems);

HRESULT WINAPI PeerGraphEndEnumeration(
                IN  HPEERENUM   hPeerEnum);

HRESULT WINAPI PeerGraphCreate(
                IN  PPEER_GRAPH_PROPERTIES   pGraphProperties,
                IN  PCWSTR      pwzDatabaseName,
                IN  PPEER_SECURITY_INTERFACE
pSecurityInterface,
                OUT PHGRAPH     phGraph);

HRESULT WINAPI PeerGraphOpen(
                IN  PCWSTR      pwzGraphId,
                IN  PCWSTR      pwzPeerId,
                IN  PCWSTR      pwzDatabaseName,
                IN  PPEER_SECURITY_INTERFACE
pSecurityInterface,
                IN  ULONG       cRecordTypeSyncPrecedence,
                IN  GUID *      pRecordTypeSyncPrecedence,
                OUT PHGRAPH     phGraph);

HRESULT WINAPI PeerGraphListen(
                IN  HGRAPH      hGraph,
                IN  DWORD       dwScope,
                IN  DWORD       dwScopeId,
                IN  WORD        wPort);

HRESULT WINAPI PeerGraphConnect(
                IN  HGRAPH      hGraph,
                IN  PCWSTR      pwzPeerId,
                IN  PPEER_ADDRESS pAddress,
                OUT ULONGLONG * pullConnectionId);

HRESULT WINAPI PeerGraphClose(
                IN  HGRAPH      hGraph);

HRESULT WINAPI PeerGraphDelete(
                IN  PCWSTR      pwzGraphId,
                IN  PCWSTR      pwzPeerId,
                IN  PCWSTR      pwzDatabaseName);

HRESULT WINAPI PeerGraphGetStatus(
                IN  HGRAPH       hGraph,
                OUT DWORD      * pdwStatus);

HRESULT WINAPI PeerGraphGetProperties(
                IN  HGRAPH      hGraph,
```

```
                OUT PPEER_GRAPH_PROPERTIES *
ppGraphProperties);

HRESULT WINAPI PeerGraphSetProperties(
                IN  HGRAPH      hGraph,
                IN  PPEER_GRAPH_PROPERTIES pGraphProperties);

// Eventing interfaces

HRESULT WINAPI PeerGraphRegisterEvent(
                IN  HGRAPH       hGraph,
                IN  HANDLE       hEvent,
                IN  ULONG        cEventRegistrations,
                IN  PPEER_GRAPH_EVENT_REGISTRATION
pEventRegistrations,
                OUT HPEEREVENT * phPeerEvent);

HRESULT WINAPI PeerGraphUnregisterEvent(
                IN  HPEEREVENT   hPeerEvent);

HRESULT WINAPI PeerGraphGetEventData(
                IN  HPEEREVENT   hPeerEvent,
                OUT PPEER_GRAPH_EVENT_DATA * ppEventData);


// Data Storage

HRESULT WINAPI PeerGraphGetRecord(
                IN  HGRAPH       hGraph,
                IN  GUID *       pRecordId,
                OUT PPEER_RECORD * ppRecord);

HRESULT WINAPI PeerGraphAddRecord(
                IN  HGRAPH       hGraph,
                IN  PPEER_RECORD pRecord,
                OUT GUID       * pRecordId);

HRESULT WINAPI PeerGraphUpdateRecord(
                IN  HGRAPH       hGraph,
                IN  PPEER_RECORD pRecord);

HRESULT WINAPI PeerGraphDeleteRecord(
                IN  HGRAPH       hGraph,
                IN  GUID *       pRecordId,
                IN  BOOL         fLocal);

HRESULT WINAPI PeerGraphEnumRecords(
                IN  HGRAPH       hGraph,
                IN  GUID *       pRecordType,
                IN  PCWSTR       pwzPeerId,
                OUT HPEERENUM *  phPeerEnum);

HRESULT WINAPI PeerGraphSearchRecords(
                IN  HGRAPH       hGraph,
                IN  PCWSTR       pwzCriteria,
                OUT HPEERENUM  * phPeerEnum);
```

```
HRESULT WINAPI PeerGraphExportDatabase(
                IN  HGRAPH       hGraph,
                IN  PCWSTR       pwzFilePath);

HRESULT WINAPI PeerGraphImportDatabase(
                IN  HGRAPH       hGraph,
                IN  PCWSTR       pwzFilePath);

HRESULT WINAPI PeerGraphValidateDeferredRecords(
                IN  HGRAPH       hGraph,
                IN  ULONG        cRecordIds,
                IN  GUID *       pRecordIds);


// Node/Connection interfaces

HRESULT WINAPI PeerGraphOpenDirectConnection(
                IN  HGRAPH       hGraph,
                IN  PCWSTR       pwzPeerId,
                IN  PPEER_ADDRESS pAddress,
                OUT ULONGLONG *  pullConnectionId);

HRESULT WINAPI PeerGraphSendData(
                IN  HGRAPH       hGraph,
                IN  ULONGLONG    ullConnectionId,
                IN  GUID *       pType,
                IN  ULONG        cbData,
                IN  PVOID        pvData);

HRESULT WINAPI PeerGraphCloseDirectConnection(
                IN  HGRAPH       hGraph,
                IN  ULONGLONG    ullConnectionId);

HRESULT WINAPI PeerGraphEnumConnections(
                IN  HGRAPH       hGraph,
                IN  DWORD        dwFlags,         //
PEER_CONNECTION_FLAGS
                OUT HPEERENUM  * phPeerEnum);

HRESULT WINAPI PeerGraphEnumNodes(
                IN  HGRAPH       hGraph,
                IN  PCWSTR       pwzPeerId,
                OUT HPEERENUM  * phPeerEnum);

HRESULT WINAPI PeerGraphSetPresence(
                IN  HGRAPH       hGraph,
                IN  BOOL         fPresent);

HRESULT WINAPI PeerGraphGetNodeInfo(
                IN  HGRAPH       hGraph,
                IN  ULONGLONG    ullNodeId,
                OUT PPEER_NODE_INFO * ppNodeInfo);

HRESULT WINAPI PeerGraphSetNodeAttributes(
                IN  HGRAPH       hGraph,
                IN  PCWSTR       pwzAttributes);
```

```
HRESULT WINAPI PeerGraphSystemTimeFromGraphTime(
                IN  HGRAPH      hGraph,
                IN  FILETIME *  pftGraphTime,
                OUT FILETIME *  pftSystemTime);
```

[0032] In one embodiment of the grouping component 312 employed by the present invention, interfaces and methods can be implemented through an API. Such an embodiment is particularly well suited for the Microsoft Windows XP operating system in which the APIs may be as follows:

```
HRESULT WINAPI PeerGroupStartup(
                IN  WORD          wVersionRequested,
                OUT PPEER_VERSION_DATA pVersionData);

HRESULT WINAPI PeerGroupShutdown();

VOID WINAPI PeerFreeData(
                IN  PVOID         pvData);

HRESULT WINAPI PeerGetItemCount(
                IN  HPEERENUM     hPeerEnum,
                OUT PULONG        pCount);

HRESULT WINAPI PeerGetNextItem(
                IN  HPEERENUM     hPeerEnum,
                IN OUT PULONG     pCount,
                OUT PVOID       * ppvItems);

HRESULT WINAPI PeerEndEnumeration(
                IN  HPEERENUM     hPeerEnum);


//////////////////////////////////////////////////
// Group interfaces

HRESULT WINAPI PeerGroupCreate(
                IN  PPEER_GROUP_PROPERTIES pProperties,
                OUT HGROUP      * phGroup);

HRESULT WINAPI PeerGroupOpen(
                IN  PCWSTR        pwzIdentity,
                IN  PCWSTR        pwzGroupPeerName,
                IN  PCWSTR        pwzCloud,
                OUT HGROUP      * phGroup);

HRESULT WINAPI PeerGroupJoin(
                IN  PCWSTR        pwzIdentity,
                IN  PCWSTR        pwzInvitation,
                IN  PCWSTR        pwzCloud,
                OUT HGROUP      * phGroup);

HRESULT WINAPI PeerGroupConnect(
```

```
                IN  HGROUP        hGroup);

HRESULT WINAPI PeerGroupClose(
                IN  HGROUP        hGroup);

HRESULT WINAPI PeerGroupDelete(
                IN  PCWSTR        pwzIdentity,
                IN  PCWSTR        pwzGroupPeerName);

HRESULT WINAPI PeerGroupCreateInvitation(
                IN  HGROUP        hGroup,
                IN  PCWSTR        pwzIdentityInfo,
                IN  FILETIME *    pftExpiration,
                IN  ULONG         cRoles,
                IN  PEER_ROLE_ID*   pRoles,
                OUT PWSTR       * ppwzInvitation);

HRESULT WINAPI PeerGroupParseInvitation(
                IN  PCWSTR        pwzInvitation,
                OUT PPEER_INVITATION_INFO * ppInvitationInfo);

HRESULT WINAPI PeerGroupGetStatus(
                IN  HGROUP        hGroup,
                OUT DWORD       * pdwStatus);

HRESULT WINAPI PeerGroupGetProperties(
                IN  HGROUP        hGroup,
                OUT PPEER_GROUP_PROPERTIES * ppProperties);

HRESULT WINAPI PeerGroupSetProperties(
                IN  HGROUP        hGroup,
                IN  PPEER_GROUP_PROPERTIES pProperties);

HRESULT WINAPI PeerGroupEnumMembers(
                IN  HGROUP        hGroup,
                IN  DWORD         dwFlags,      //
PEER_MEMBER_FLAGS
                IN  PCWSTR        pwzIdentity,
                OUT HPEERENUM   * phPeerEnum);

HRESULT WINAPI PeerGroupOpenDirectConnection(
                IN  HGROUP        hGroup,
                IN  PCWSTR        pwzIdentity,
                IN  PPEER_ADDRESS pAddress,
                OUT ULONGLONG   * pullConnectionId);

HRESULT WINAPI PeerGroupCloseDirectConnection(
                IN  HGROUP        hGroup,
                IN  ULONGLONG     ullConnectionId);

HRESULT WINAPI PeerGroupEnumConnections(
                IN  HGROUP        hGroup,
                IN  DWORD         dwFlags,         //
PEER_CONNECTION_FLAGS
                OUT HPEERENUM   * phPeerEnum);

HRESULT WINAPI PeerGroupSendData(
```

```
                IN  HGROUP          hGroup,
                IN  ULONGLONG       ullConnectionId,
                IN  GUID *          pType,
                IN  ULONG           cbData,
                IN  PVOID           pvData);


// Eventing interfaces

HRESULT WINAPI PeerGroupRegisterEvent(
                IN  HGROUP          hGroup,
                IN  HANDLE          hEvent,
                IN  DWORD           cEventRegistration,
                IN  PPEER_GROUP_EVENT_REGISTRATION
pEventRegistrations,
                OUT HPEEREVENT * phPeerEvent);

HRESULT WINAPI PeerGroupUnregisterEvent(
                IN  HPEEREVENT      hPeerEvent);

HRESULT WINAPI PeerGroupGetEventData(
                IN  HPEEREVENT      hPeerEvent,
                OUT PPEER_GROUP_EVENT_DATA * ppEventData);


// Data Storage

HRESULT WINAPI PeerGroupGetRecord(
                IN  HGROUP          hGroup,
                IN  GUID *          pRecordId,
                OUT PPEER_RECORD * ppRecord);

HRESULT WINAPI PeerGroupAddRecord(
                IN  HGROUP          hGroup,
                IN  PPEER_RECORD pRecord,
                OUT GUID          * pRecordId);

HRESULT WINAPI PeerGroupUpdateRecord(
                IN  HGROUP          hGroup,
                IN  PPEER_RECORD pRecord);

HRESULT WINAPI PeerGroupDeleteRecord(
                IN  HGROUP          hGroup,
                IN  GUID *          pRecordId);

HRESULT WINAPI PeerGroupEnumRecords(
                IN  HGROUP          hGroup,
                IN  GUID *          pRecordType,
                OUT HPEERENUM    * phPeerEnum);

HRESULT WINAPI PeerGroupSearchRecords(
                IN  HGROUP          hGroup,
                IN  PCWSTR          pwzCriteria,
                OUT HPEERENUM    * phPeerEnum);

HRESULT WINAPI PeerGroupExportDatabase(
                IN  HGROUP          hGroup,
```

```
                    IN  PCWSTR          pwzFilePath);

HRESULT WINAPI PeerGroupImportDatabase(
                    IN  HGROUP          hGroup,
                    IN  PCWSTR          pwzFilePath);

HRESULT WINAPI PeerGroupAuthorizeMembership(
                    IN  HGROUP          hGroup,
                    IN  PPEER_MEMBERSHIP_INFO pMemberInfo,
                    IN  BOOL            fAuthorize);

HRESULT WINAPI PeerGroupPeerTimeToUniversalTime(
                    IN  HGROUP          hGroup,
                    IN  FILETIME *      pftPeerTime,
                    OUT FILETIME *      pftUniversalTime);

HRESULT WINAPI PeerGroupUniversalTimeToPeerTime(
                    IN  HGROUP          hGroup,
                    IN  FILETIME *      pftUniversalTime,
                    OUT FILETIME *      pftPeerTime);
```

[0033] Figures 4 and 5 illustrate the peer-to-peer group creation and joining process. Beginning with step 500, the group creator 100 requests that a peer identity for the application be created by the identity manager 310. Next, in step 502, the group creator 100 requests that a group 404 be created. In step 504 a group joiner 100 requests that a peer identity be created by the identity manager 310. Next, in step 506, using the discovery protocol 308 the group joiner 100 locates a group 404 to join. In step 508 the group joiner 100 opens the group 404 that the joiner wishes to join and in step 510 the group joiner 100 sends its identity 400 credentials to the group creator 100. Upon receipt of the group joiner's identity credentials 400 by the group creator 100, in step 512, the group creator 100 requests that an invitation 402 be created to send to the group joiner 100. Next, in step 514, the group creator 100 sends the invitation 402 to the group joiner 100. Upon receipt of the invitation 402 in step 516, the group joiner 100 accepts the invitation 402 and joins the group 404 in step 518. The preceding steps can be repeated resulting in the formation of a peer-to-peer group such as the one illustrated in Figure 6.

IV. Content Distribution Leveraging a Peer-to-peer Network Infrastructure

[0034] With reference to Figure 6, an exemplary peer-to-peer group is illustrated. In the group, each peer-to-peer node 100 is reachable via a path on the graph of the group. Each peer node 100 in the group has an instance of the replicated store 314. The replicated store 314 is associated with a graph or a group and maintains metadata as to the current state of each node. When a node 100 connects to a group it first synchronizes the replicated store database 314. The nodes 100 maintain this database automatically.

[0035] The replicated store 314 houses metadata about the peer-to-peer group in the form of records 600 residing in the store 314. Each record can contain a record ID field, a record type field, and an attribute field. In the case of the present invention, the metadata in the store reflects what content has been distributed to nodes 100 of the group. For each piece of content that has been distributed to a node 100 in the group, a record 600 corresponding to that piece of content exists in the replicated store 314. This record 600 possesses a location attribute that enables a node 100 in the group to ascertain if the desired content is available from within the peer group and, if so, from which nodes 100 in the group.

[0036] Turning to Figure 7, the method of peer-to-peer content distribution leveraging a peer-to-peer networking infrastructure is illustrated. Beginning with step 700, a content distribution server publishes content for distribution. Continuing with step 702, a client computing device makes a request to the content distribution server for the published content. In step 704, the content distribution server sends the content to the client computing device and, in step 706, the client computing device receives the desired

content. Next, in step 708, the client computing device updates the instance of the replicated store located on the client computing device to reflect the content that was obtained in step 706. In step 710, the update to the replicated store is propagated through the peer-to-peer group to the instances of the replicated store on the nodes of the group. Next, in step 712, a node desiring a piece of content can consult the replicated store to determine if the content is obtainable via the peer-to-peer group. Finally, in step 714, a node which has located the content it desires can receive that content from the peer node in the group having the desired content.

[0037] The above described method can be applied to a variety of content distribution scenarios. One such scenario is illustrated in Figure 8. Figure 8 depicts a residential network connected to the Internet 204 via a shared broadband or dial-up connection 208. In a connectivity scheme such as this one, all of the computing devices in the residential network share the bandwidth of the connection 208. In this scenario, leveraging the peer-to-peer networking infrastructure in distributing content would result in a savings of bandwidth of the shared connection 208. Under a server-based content distribution scheme, a home network having five computers desiring content (e.g., applications, patches, product upgrades, virus definitions, etc.) from a central server would require that each computer individually obtain the content. This necessitates five separate requests for content from the central server 206, each request drawing on bandwidth of the connection 208. As illustrated in Figure 8, the home network can form a peer-to-peer group 404 of computing devices and one client computing device can download the desired content from the external content server 206 via the Internet 204 and connection 208. The other peer computing devices can then simply obtain the content from the node

that downloaded it from the external content server 206. The scenario can also be extended for use with applications such as distributed web caches wherein clients can download a web content from another client close to it who may have downloaded that web content recently. Since the speed of the local network connections will typically far rival that of the broadband or dial-up connection, this results in a better overall experience for the user.

[0038] An additional scenario in which the leveraging of the peer-to-peer network infrastructure results in a more efficient distribution of content is illustrated in Figure 9. Suppose, for example, the external content server 206 of Figure 9 is a virus definition server for a large antivirus company. As will be appreciated by one of ordinary skill in the art, the virus definition development life-cycle is typically characterized as a very short "ship cycle." As soon as a new antivirus definition file is available, it is shipped and/or made available for download to the antivirus subscribers. The frequency of such updates to antivirus programs is quite high. Under a server-based content distribution scenario, these updates would be hosted on the external content server 206. The spikes in downloading of these updates, when posted, could potentially cause scaling issues. By leveraging the peer-to-peer networking infrastructure in distributing content, such as virus updates, load on the external content server 206 can be reduced. As illustrated in Figure 9, a content distribution group 404 can be created and managed by the external content server 206 and requests for content from computing devices 100 can be redirected from the server 206 to the content distribution group 404. Thus, only a handful of customers would need to connect up to the main anti-virus server 206 and the virus patch would automatically be propagated through the content distribution group. In

this scenario, the patch itself would likely be signed by the company to ensure that no customer in the distribution group 404 could spoof the patch as coming from the company.

[0039] Yet another scenario in which the leveraging of the peer-to-peer network infrastructure can result in a more efficient distribution of content is illustrated in Figure 10. In a large network such as an enterprise installation, the speed of dissemination of content can be improved when only one or two computing devices 100 download desired content and then, in turn, deliver the content to other computing devices 100. In addition, the computing devices 100 which receive the content from those computing devices 100 that downloaded the content from the content server 206 can advertise the presence of the content such that the load on individual computing devices gets spread out over time. This scenario lends itself well to large enterprise installations where it is not unusual for there to be a need for all computing devices in remote branch offices to download software from a remote server located across a slow wide area network (WAN) link. In much the same respect as the peer-to-peer content distribution solution for a residential network in Figure 8, the speed of the local network connections will typically be superior to that of the WAN connection and thus result in a more efficient distribution of content. Additionally, in much the same way that the peer-to-peer content distribution solution of Figure 9 increases the speed at which content can be disseminated over a large area such as the Internet, the peer-to-peer content distribution scenario of Figure 10 can be particularly useful in situations where the dissemination of the content is time-sensitive, such as a presentation file for an on-line company meeting or a streaming audio file for an on-line concert performance.

[0040] In view of the many possible embodiments to which the principles of this invention may be applied, it should be recognized that the embodiments described herein with respect to the drawing figures are meant to be illustrative only and should not be taken as limiting the scope of invention. For example, for performance reasons the method of the present invention may be implemented in hardware, rather than in software. Therefore, the invention as described herein contemplates all such embodiments as may come within the scope of the following claims and equivalents thereof.